EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	72	erik near johnson.in.	US-PGPUB; USPAT	OR	ON	2007/01/30 16:19
S2	28	james near jason.in.	US-PGPUB; USPAT	OR	ON	2007/01/30 16:19
S3	8	harrick near vin.in.	US-PGPUB; USPAT	OR	ON	2007/01/30 16:20
S4	15110	intel.as.	US-PGPUB; USPAT	OR	ON	2007/01/30 16:20
S5	168	S4 and (thread and instruction and process$4).clm.	US-PGPUB; USPAT	OR	ON	2007/01/30 16:20
S6	10	S5 and (insert$4 and processor).clm.	US-PGPUB; USPAT	OR	ON	2007/01/30 16:25
S8	83	S4 and (thread with instruction with processor).clm.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/30 16:26
S9	98	("5659701" "4837688" "5481719" "5590356" "5606696" "5611073" "5634022" "5752068" "5933627" "6006320" "4782441" "4993017" "5404523" "5485626" "5584023" "5754759" "5826090" "5958049" "6188411" "4777589" "4803622" "4951195" "4965721" "5008812" "5278973" "5301302" "5321744" "5574892" "5598409" "5619409" "5636124" "5751797" "5870614" "5906002" "5968147" "6000029" "6006028" "6009454" "6047351" "6129458" "6131186" "6199181" "6237089" "6272481" "6295600" "6338063" "6420903" "6578137" "6785887" "7055142").pn.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/30 16:37
S10	63	(insert$4 near instruction) with thread	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/30 16:57
S11	1552	thread adj chang$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/30 17:08

EAST Search History

S12	33	S11 and (insert$4 near instruction)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/30 17:08
S13	22	("3771138" | "5361337" | "5386561" | "5392437" | "5553291" | "5586332" | "5630130" | "5761522" | "5787297" | "5809522" | "5892959" | "5968160" | "5983339" | "5996085" | "6009454" | "6052708" | "6085218" | "6088788" | "6212544" | "6256775" | "6289461" | "6314530").PN.	US-PGPUB; USPAT; USOCR	OR	ON	2007/01/30 17:22
S14	2513	(thread adj (chang$4 or switch$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:16
S15	225	S14 and ((insert$4 or add$3) near3 instruction)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 08:08
S16	213	S15 and processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 08:08
S17	28	S16 and multi$1task$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 08:09
S18	85	("4937783" | "4974154" | "4974155" | "5040107" | "5165025" | "5179702" | "5230068" | "5247675" | "5261097" | "5265213" | "5287467" | "5297281" | "5353418" | "5353419" | "5404469" | "5421022" | "5430851" | "5471593" | "5499349" | "5511192" | "5560029").PN. OR ("5812811"). URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2007/01/31 08:22

EAST Search History

S20	667	(pre$1emptive or co$1operat$4) near multi$1task$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 08:39
S21	11	S14 and S20	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 08:39
S22	2	average near3 (consecutive adj instruction)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:15
S23	1556	average near3 instruction	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:15
S24	212	S23 and thread	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:16
S25	201	S24 and processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:16
S26	25	S25 and (thread adj (chang$4 or switch$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:16
S28	158	(standard adj deviation) with instruction	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:22

EAST Search History

S29	21	S28 and thread and processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:22
S30	975	graph near3 instruction	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:28
S31	426	S30 and (graph with node)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:28
S32	68	S31 and thread and processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 10:28
S33	195	co$1operative adj multi$1task$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 12:01
S34	2	co$1operative adj multi$1task$4 adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/31 12:01

Terms used **thread switch insert instruction** Found **28,435** of **196,064**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Effective thread management on network processors with compiler analysis
Xiaotong Zhuang, Santosh Pande
June 2006 **ACM SIGPLAN Notices , Proceedings of the 2006 ACM SIGPLAN/SIGBED conference on Language, compilers and tool support for embedded systems LCTES '06**, Volume 41 Issue 7
Publisher: ACM Press
Full text available: pdf(466.13 KB) Additional Information: full citation, abstract, references, index terms

Mapping packet processing tasks on network processor micro-engines involves complex tradeoffs that relating to maximizing parallelism and pipelining. Due to an increase in the size of the code store and complexity of the application requirements, network processors are being programmed with heterogeneous threads that may execute code belonging to different tasks on a given micro-engine. Also, most network applications are streaming applications that are typically processed in a pipelined fashion ...

Keywords: CPU scheduling, compiler optimizations, network processors, real-time scheduling

2 Balancing register allocation across threads for a multithreaded network processor
Xiaotong Zhuang, Santosh Pande
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation PLDI '04**, Volume 39 Issue 6
Publisher: ACM Press
Full text available: pdf(429.85 KB) Additional Information: full citation, abstract, references, citings, index terms

Modern network processors employ multi-threading to allow concurrency amongst multiple packet processing tasks. We studied the properties of applications running on the network processors and observed that their imbalanced register requirements across different threads at different program points could lead to poor performance. Many times application needs demand some threads to be more performance critical than others and thus by controlling the register allocation across threads one could impa ...

Keywords: multithreaded processor, network processor, register allocation

3 Software thread integration for embedded system display applications
Alexander G. Dean
February 2006 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 5 Issue 1
Publisher: ACM Press

Full text available: pdf(1.40 MB) Additional Information: full citation, abstract, references, index terms

Embedded systems require control of many concurrent real-time activities, leading to system designs that feature a variety of hardware peripherals, with each providing a specific, dedicated service. These peripherals increase system size, cost, weight, and design time. Software thread integration (STI) provides low-cost thread concurrency on general-purpose processors by automatically interleaving multiple threads of control into one. This simplifies hardware to software migration (which elimina ...

Keywords: fine-grain concurrency, hardware to software migration, software thread integration

4 Improving instruction cache performance in OLTP

Stavros Harizopoulos, Anastassia Ailamaki

September 2006 **ACM Transactions on Database Systems (TODS)**, Volume 31 Issue 3

Publisher: ACM Press

Full text available: pdf(783.16 KB) Additional Information: full citation, abstract, references, index terms

Instruction-cache misses account for up to 40% of execution time in online transaction processing (OLTP) database workloads. In contrast to data cache misses, instruction misses cannot be overlapped with out-of-order execution. Chip design limitations do not allow increases in the size or associativity of instruction caches that would help reduce misses. On the contrary, the effective instruction cache size is expected to further decrease with the adoption of multicore and multithreading ...

Keywords: Instruction cache, cache misses

5 An elementary processor architecture with simultaneous instruction issuing from multiple threads

Hiroaki Hirata, Kozo Kimura, Satoshi Nagamine, Yoshiyuki Mochizuki, Akio Nishimura, Yoshimori Nakase, Teiji Nishizawa

April 1992 **ACM SIGARCH Computer Architecture News , Proceedings of the 19th annual international symposium on Computer architecture ISCA '92**, Volume 20 Issue 2

Publisher: ACM Press

Full text available: pdf(1.03 MB) Additional Information: full citation, abstract, references, citings, index terms

In this paper, we propose a multithreaded processor architecture which improves machine throughput. In our processor architecture, instructions from different threads (not a single thread) are issued simultaneously to multiple functional units, and these instructions can begin execution unless there are functional unit conflicts. This parallel execution scheme greatly improves the utilization of the functional unit. Simulation results show that by executing two and four threads in parallel ...

6 Whole-program optimization for time and space efficient threads

Dirk Grunwald, Richard Neves

September 1996 **ACM SIGPLAN Notices , ACM SIGOPS Operating Systems Review , Proceedings of the seventh international conference on Architectural support for programming languages and operating systems ASPLOS-VII**, Volume 31 , 30 Issue 9 , 5

Publisher: ACM Press

Full text available: pdf(1.11 MB) Additional Information: full citation, abstract, references, citings, index terms

Modern languages and operating systems often encourage programmers to use *threads,* or independent control streams, to mask the overhead of some operations and simplify program structure. Multitasking operating systems use threads to mask communication latency, either with hardwares devices or users. Client-server applications typically use threads to simplify the complex control-flow that arises when multiple clients are used.

Recently, the scientific computing community has started using ...

7 Multithreading and value prediction: Correctly implementing value prediction in microprocessors that support multithreading or multiprocessing

Milo M. K. Martin, Daniel J. Sorin, Harold W. Cain, Mark D. Hill, Mikko H. Lipasti

December 2001 **Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture MICRO 34**

Publisher: IEEE Computer Society

Full text available: pdf(1.04 MB) Additional Information: full citation, abstract, references, citings

Publisher Site

This paper explores the interaction of value prediction with thread-level parallelism techniques, including multithreading and multiprocessing, where correctness is defined by a memory consistency model. Value prediction subtly interacts with the memory consistency model by allowing data dependent instructions to be reordered. We find that predicting a value and later verifying that the value eventually calculated is the same as the value predicted is not always sufficient.We present an example ...

8 Compiler analysis and optimization: Providing time- and space- efficient procedure calls for asynchronous software thread integration

Vasanth Asokan, Alexander G. Dean

September 2004 **Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems CASES '04**

Publisher: ACM Press

Full text available: pdf(289.56 KB) Additional Information: full citation, abstract, references, citings, index terms

Asynchronous Software Thread Integration (ASTI) provides fine-grain concurrency in real-time threads by statically scheduling (integrating) code from primary threads into secondary threads, reducing the context switching needed and allowing recovery of fine-grain idle time. Unlike STI, ASTI allows asynchronous thread progress.Current ASTI techniques do not support procedure calls in the secondary thread because they lead to timing conflicts during static scheduling. ASTI requires knowing the sec ...

Keywords: asynchronous software thread integration, fine-grain concurrency, hardware to software migration, software-implemented communication protocol controllers

9 Implementing an efficient vector instruction set in a chip multi-processor using micro-threaded pipelines

Chris Jesshope

January 2001 **Australian Computer Science Communications , Proceedings of the 6th Australasian conference on Computer systems architecture ACSAC '01**, Volume 23 Issue 4

Publisher: IEEE Computer Society, IEEE Computer Society Press

Full text available: pdf(911.26 KB) Additional Information: full citation, abstract, references, citings

Publisher Site

This paper looks at a combination of two techniques, one of which, using a vector instruction set, has a long history dating back to pipelined vector supercomputers, such as the Cray 1 and its successors. The other technique, multi-threading, is also well understood. The novel approach proposed in this paper combines both vertical and horizontal micro-threading with vector instruction descriptors. It will be shown that a family of threads can represent a vector instruction with dependencies betw ...

10 New garbage collection algorithms and strategies: Dynamic selection of application-specific garbage collectors

Sunil Soman, Chandra Krintz, David F. Bacon

October 2004 **Proceedings of the 4th international symposium on Memory management ISMM '04**

Publisher: ACM Press

Full text available: pdf(185.74 KB) Additional Information: full citation, abstract, references, citings, index terms

Much prior work has shown that the performance enabled by garbage collection (GC) systems is highly dependent upon the behavior of the application as well as on the available resources. That is, no single GC enables the best performance for all programs and all heap sizes. To address this limitation, we present the design, implementation, and empirical evaluation of a novel Java Virtual Machine (JVM) extension that facilitates dynamic switching between a number of very different and popular g ...

Keywords: Java, annotation, application-specific collection, dynamic selection, hot-swapping, virtual machine

11 WBIA'05: Controlling program execution through binary instrumentation

Heidi Pan, Krste Asanović, Robert Cohn, Chi-Keung Luk

December 2005 **ACM SIGARCH Computer Architecture News**, Volume 33 Issue 5

Publisher: ACM Press

Full text available: pdf(261.35 KB) Additional Information: full citation, abstract, references, index terms

Binary instrumentation has been widely used to observe dynamic program behavior, but current binary instrumentation systems do not allow the tool writer to alter the program execution path. This paper introduces some simple and general mechanisms for a binary instrumentation infrastructure to provide control over the application's execution path, allowing tools to replay or skip parts of the application, and to start or switch between threads. Specifically, the technique provides the following t ...

12 Compiler-based prefetching for recursive data structures

Chi-Keung Luk, Todd C. Mowry

October 1996 **ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , Proceedings of the seventh international conference on Architectural support for programming languages and operating systems ASPLOS-VII**, Volume 30 , 31 Issue 5 , 9

Publisher: ACM Press

Full text available: pdf(1.51 MB) Additional Information: full citation, abstract, references, citings, index terms

Software-controlled data prefetching offers the potential for bridging the ever-increasing speed gap between the memory subsystem and today's high-performance processors. While prefetching has enjoyed considerable success in array-based numeric codes, its potential in pointer-based applications has remained largely unexplored. This paper investigates compiler-based prefetching for pointer-based applications---in particular, those containing recursive data structures. We identify the fundamental ...

13 Multithreaded Extension to Multicluster VLIW Processors for Embedded Applications

Domenico Barretta, William Fornaciari, Mariagiovanna Sami, Daniele Bagni

March 2005 **Proceedings of the conference on Design, Automation and Test in Europe - Volume 2 DATE '05**

Publisher: IEEE Computer Society

Full text available: pdf(123.29 KB) Additional Information: full citation, abstract, index terms

Instruction Level Parallelism (ILP) extraction for multi-cluster VLIW processors is a very hard task. In this paper, we propose a retargetable architecture that can exploit ILP and thread level parallelism jointly, thus allowing an easier parallelism extraction and improving the performance with respect to traditional multicluster VLIW processors.

14 Opportunistic Transient-Fault Detection

Mohamed A. Gomaa, T. N. Vijaykumar

May 2005 **ACM SIGARCH Computer Architecture News , Proceedings of the 32nd Annual International Symposium on Computer Architecture ISCA '05**,

Volume 33 Issue 2
Publisher: IEEE Computer Society, ACM Press
Full text available: pdf(150.01 KB) Additional Information: full citation, abstract, citings, index terms

CMOS scaling increases susceptibility of microprocessors to transient faults. Most current proposals for transient-fault detection use full redundancy to achieve perfect coverage while incurring significant performance degradation. However, most commodity systems do not need or provide perfect coverage. A recent paper explores this leniency to reduce the soft-error rate of the issue queue during L2 misses while incurring minimal performance degradation. Whereas the previous paper reduces soft-er ...

15 Testing and analysis: Demand-driven structural testing with dynamic instrumentation

Jonathan Misurda, James A. Clause, Juliya L. Reed, Bruce R. Childers, Mary Lou Soffa
May 2005 **Proceedings of the 27th international conference on Software engineering ICSE '05 , Proceedings of the 27th international conference on Software engineering ICSE '05**
Publisher: ACM Press, IEEE Computer Society
Full text available: pdf(180.97 KB) Additional Information: full citation, abstract, references, citings, index terms
Publisher Site

Producing reliable and robust software has become one of the most important software development concerns in recent years. Testing is a process by which software quality can be assured through the collection of information. While testing can improve software reliability, current tools typically are inflexible and have high over-heads, making it challenging to test large software projects. In this paper, we describe a new scalable and flexible framework for testing programs with a novel demand-dr ...

Keywords: Java programming language, code coverage, demand-driven instrumentation, structural testing, testing

16 Fine-grain multithreading with the EM-X multiprocessor

Andrew Sohn, Yuetsu Kodama, Jui Ku, Mitsuhisa Sato, Hirofumi Sakane, Hayato Yamana, Shuichi Sakai, Yoshinori Yamaguchi
June 1997 **Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures SPAA '97**
Publisher: ACM Press
Full text available: pdf(1.50 MB) Additional Information: full citation, references, citings, index terms

17 Asynchronous software thread integration for efficient software

Nagendra J. Kumar, Siddhartha Shivshankar, Alexander G. Dean
June 2004 **ACM SIGPLAN Notices , Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems LCTES '04**, Volume 39 Issue 7
Publisher: ACM Press
Full text available: pdf(229.95 KB) Additional Information: full citation, abstract, references, citings, index terms

The overhead of context-switching limits efficient scheduling of multiple concurrent threads on a uniprocessor when real-time requirements exist. Existing software thread integration (STI) methods reduce context switches, but only provide synchronous thread progress within integrated functions. For the remaining, non-integrated portions of the secondary threads to run and avoid starvation, the primary thread must have adequate amounts of coarse-grain idle time (longer than two context-switches). ...

Keywords: J1850, asynchronous software thread integration, fine-grain concurrency, hardware to software migration, software-implemented communication protocol controllers

18 Responsiveness without interrupts

Dejan Perkovic, Peter J. Keleher
May 1999 **Proceedings of the 13th international conference on Supercomputing ICS '99**
Publisher: ACM Press
Full text available: pdf(1.20 MB) Additional Information: full citation, references, citings, index terms

19 Software-controlled fault tolerance

George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, David I. August, Shubhendu S. Mukherjee
December 2005 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 2 Issue 4
Publisher: ACM Press
Full text available: pdf(638.90 KB) Additional Information: full citation, abstract, references, citings, index terms

Traditional fault-tolerance techniques typically utilize resources ineffectively because they cannot adapt to the changing reliability and performance demands of a system. This paper proposes software-controlled fault tolerance, a concept allowing designers and users to tailor their performance and reliability for each situation. Several software-controllable fault-detection techniques are then presented: SWIFT, a software-only technique, and CRAFT, a suite of hybrid hardware/software techniques ...

Keywords: Software-controlled fault tolerance, fault detection, reliability

20 Tolerating latency in multiprocessors through compiler-inserted prefetching

Todd C. Mowry
February 1998 **ACM Transactions on Computer Systems (TOCS)**, Volume 16 Issue 1
Publisher: ACM Press
Full text available: pdf(410.70 KB) Additional Information: full citation, abstract, references, citings, index terms, review

The large latency of memory accesses in large-scale shared-memory multiprocessors is a key obstacle to achieving high processor utilization. Software-controlled prefetching is a technique for tolerating memory latency by explicitly executing instructions to move data close to the processor before the data are actually needed. To minimize the burden on the programmer, compiler support is needed to automatically insert prefetch instructions into the code. A key challenge when ...

Keywords: compiler optimization, prefetching

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next